INTERASM

InterAsm

Professionel dansk assembler til Amiga 500, 1000, 2000 og 3000.

Program og manual skrevet af Kenneth Bernholm

INTERACTIVISION

Hovedgaardsvej 4, DK-8600 Silkeborg

PRODUKTIONS-OPLYSNINGER SAMT COPYRIGHT-BESTEMMELSER

InterAsm til Amiga er skrevet af Kenneth Bernholm.
Manualen er skrevet og redigeret af Kenneth Bernholm.
InterAsm er produceret og udgivet af Interactivision Aps.

InterAsm er copyright (C) 1990 Interactivision ApS.
Alle rettigheder reserveret.
Denne publikation (disketter, manual og emballage) må hverken helt eller delvist reproduceres, overføres, genskrives, gemmes på et system, hvorfra stoffet senere kan genbruges, eller oversættes til noget sprog uden skriftlig tilladelse fra Interactivision ApS.

Navnene Amiga, Workbench og Kickstart er varemærker tilhørende Commodore Busines Machines Inc.
Navnene InterAsm samt Interactivision logoet er varemærker tilhørende Interactivision ApS.

Systemfaciliteter og redskaber er programmer under copyright fra Commodore Busines Machines Inc., der bruges under licens af Interactivision ApS. og kun i forbindelse med InterAsm. Commodore software bør ikke kopieres over på anden disk (undtagen af arkiveringshensyn) eller ind i hukommelsen, medmindre det sker i forbindelse med brug af InterAsm. Når InterAsm er færdig med jobudførelsen bør Commodore software'n ikke bruges af andre programmer.

InterAsm er resultatet af mange måneders research og udvikling, hvor adskillige personer og firmaer har bidraget med viden og arbejdsindsats.
For at dette arbejde skal kunne fortsætte i fremtiden, er det nødvendigt at ulovlig kopiering af software fra Interactivision ApS. og andre lignende firmaer, ikke forekommer.

InterAsm distribueres i Europa af :

SuperSoft ApS.
Åboulevarden 51-53
DK-8000 Århus C
Tlf. 86-193244
Fax 86-201690

INDHOLDSFORTEGNELSE

	5.	- Om InterAsm
	6.	- Det kan InterAsm
KAPITEL 1. BEGYNDERENS GUIDE	7.	- Bliv registreret bruger
	7.	- Lav først en sikkerhedskopi
	7.	- Harddisk installation
	7.	- Hvad er en assembler
	8.	- Amiga'ens processor
	9.	- Hukommelsen
	10.	- Start af programmer
	11.	- Bevægelse i teksten
KAPITEL 2. GENNEMGANG AF MENUER		FILES menuen
	12.	- Load
	12.	- Save
	12.	- Clear
	12.	- Delete
	13.	- Print
	13.	- About
	13.	- Quit
		EDITOR menuen
	13.	- Mark block
	13.	- Copy block
	13.	- Cut block
	13.	- Paste block
	13.	- Search
	13.	- Replace
		ASSEMBLE menuen
	13.	- Assemble
	14.	- Run
	14.	- Disassemble
	14.	- Singlestep
	14.	- Watch

		SPECIAL menuen
	14.	- Calculator
	14.	- Show memory
	14.	- Fill memory
	14.	- Copy memory
	14.	- Search memory
	15.	- Memory info
KAPITEL 3. VIDEREGÅENDE GUIDE		OPTIONS menuen
	15.	- Editor
	15.	- Assembler
	16.	- Kommentarer
	16.	- Tekster
	16.	- Tal
	16.	- Regneudtryk
	16.	- Labels
	16.	- Konstant erklæringer
	17.	- ORG- og LOAD-direktiver
	17.	- Include filer
	17.	- Bytes, Words og Longwords
	17.	- Even og Odd
APPENDIX A	18.	- ASCII koder
APPENDIX B	22.	- Mnemonics
APPENDIX C	24.	- Fejlmeddelelser
	26.	- Nye og gamle produkter

FORORD

Kære køber,

Først og fremmest tak for at De valgte InterAsm til Deres programmeringsbehov på Amiga computeren. Dette stykke software blev udviklet ene og alene af den grund, at Interactivision som dansk softwarehus ikke var tilfreds med standarden på de p.t. på markedet værende assemblere. InterAsm blev udviklet og designet af Interactivisions programmører FOR Interactivisions programmører.
Vi håber at dette produkt vil være til lige så meget gavn for Dem, som det har været for Interactivisions programmørstab i tidens løb.

Denne manual er delt op i 3 kapitler, der stiger i sværhedsgrad, evt. i takt med brugerens udvikling. Bagest er der 5 appendices, der er ment som opslagsværk ved daglig brug af InterAsm.

KAPITEL 1.

Det første kapitel er hovedsagligt beregnet for begynderen. Her forklares om programmet og om hvordan man bruger en assembler som InterAsm. Hvis De allerede kender til Amiga assembler programmering, kan De nemt springe dette kapitel over.

KAPITEL 2.

Kapitel 2 er en gennemgang af InterAsm fra a til z. Alle menuer og vinduer beskrives, så De aldrig skal være i tvivl om de forskellige funktioners formål. Hvadenten De er begynder eller øvet bruger, bør De kigge dette kapitel igennem.

KAPITEL 3.

I kapitel 3 bliver alle InterAsm' specielle features beskrevet. For dem der ønsker at få det mest mulige ud af deres produkt, er dette kapitel en stor hjælp. Dette kapitel henvender sig hovedsagligt til den avancerede bruger, og tjener desuden som praktisk opslagsværk ved arbejde med InterAsm.

APPENDIX 1-3

De 3 appendix'es indeholder henholdsvis ASCII koder, assembler mnemonics og fejlmeddelelser.

HVAD ER INTERASM

Den første dansk udviklede assembler til Amiga, med mange nye funktioner der hidtil var forbeholdt langt dyrere maskiner. InterAsm er udviklet med henblik på at skabe et virkeligt stærkt stykke udviklingsværktøj, baseret på Amiga'ens suveræne brugerflade. Et stort arbejde er blevet lagt i at videregive så mange som muligt af de funktioner, men ellers kun finder på mini- og mainframe computere.
Alle der enten på amatør eller professionel basis, vil programmere Amiga'ens stærke processor, har brug for InterAsm. Hvad enten man ønsker at arbejde på Amiga systemets premisser, eller tilgå Amiga'ens hardware direkte, er InterAsm parat med funktioner der gør livet lettere for Amiga programmører. Arbejder man med store projekter, vil man også glæde sig over at InterAsm konstant oplyser om, hvor langt et job er kommet.

FACTS OM INTERASM

Dansk manual
Hurtig skærmopdatering
Brugerdefinerbart skærmlayout
Realtime program tracing
Avanceret resident editor
Disassemblering
Import af alle kildetekst syntax'er
Multilevel compiler med valgfri kode-optimering
Indbygget fil-pakker
100% multitasking forberedt

INTERASM ER IKKE ALENE

Et stykke software er ikke ordentlig brugbart, hvis det ikke er kompatibelt med de gængse markedstandarder. Derfor kan InterAsm naturligvis importere kildetekster fra stort set alle andre assemblere på markedet.

ANBEFALET HARDWARE

InterAsm virker på selv den mindste Amiga 500 med 512Kb RAM. Selvfølgelig kan både Amiga 1000, 2000 og 3000 også anvendes. Ekstra RAM udnyttes naturligvis. InterAsm kommer virkelig til sin ret, hvis en harddisk er tilsluttet. Læs mere om installationen inde i manualen. Bemærk at InterAsm kræver Kickstart og Workbench 1.2 eller senere version.

BLIV REGISTRERET BRUGER

I InterAsm pakken fandt De et registreringskort. Det bedes De udfylde og indsende, hvorved De lader Dem registrere som bruger af InterAsm. Denne registrering har ene og alene til formål, at garantere Dem specielt favorable priser på kommende opgraderinger af InterAsm, så De altid har et produkt der følger markedstandarden.
Husk at det er specielt vigtigt at De udfylder det område på registrerungskortet, der hedder 'Serienummer'. Hvert eksemplar af InterAsm har sit eget unikke serienummer, der findes i 'About' menuen i InterAsm. De andre punkter på registreringskortet benyttes til intern statistik, og vil naturligvis blive behandlet med største diskretion.

LAV FØRST EN SIKKERHEDSKOPI

Det første De bør gøre, inden De begynder arbejdet med InterAsm er at lave en sikkerhedskopi af disketten. InterAsm disketten er ikke kopi beskytte på nogen måde, så De kan lave Deres sikkerhedskopi ved hjælp af 'Dublicate' funktionen under Workbench. Det anbefales, at De konsulterer Deres Amiga brugervejledning, før De begiver dem i kast med kopieringen.

HARDDISK INSTALLATION

Hvis De har en harddisk tilsluttet Deres Amiga, ønsker De måske at kopiere InterAsm til harddisken. For at gøre dette, skal De i Workbench blot flytte InterAsm-ikonet fra disketten, til den skuffe på Deres hardddisk, De ønsker at opbevare InterAsm i.

HVAD ER EN ASSEMBLER

Ordet 'Assemble' betyder på dansk 'at samle'. Det er også lige præcis hvad en assembler gør. Den 'samler' den programtekst brugeren har skrevet, til noget der kaldes maskinkode. Maskinkode er ikke et læseligt programmeringsprog, men blot en samling tal, som computeren forstår som styrekoder til at påvirke elektronikken inde i maskinen.
Det er med en assembler ikke meningen, at programmøren skal kunne forstå maskinkode, men blot et sæt ordre der kaldes 'mnemonics'. Det var ikke stavet forkert - assembler-ordrer hedder faktisk 'mnemonics'. Det er måske lidt svært at udtale, men øvelse gør mester.

Disse mnemonics kan i store træk sammenlignes med ordrerne fra forskellige højniveau-sprog, såsom C, Pascal, Basic og Comal. hver enkel mnemonic kan kun een bestemt ting, og derfor kræves der adskillige mnemonics for at udføre en given operation, som et højniveau-sprog ville have som een ordre.
Et højniveau-sprog er kort fortalt en samling mere eller mindre engelsk-lignende ordrer. Hver af disse ordrer repræsenterer en samling mnemonics, som programmøren dog aldrig får at se. Når et højniveau-program skal eksekveres, bliver det ikke 'assembleret', men derimod 'kompileret'. De to ord dækker over to identiske processer, og det er altså kun på mnemonic-niveau at en programtekst 'assembleres'.
Lad os vende tilbage til de før omtalte mnemonics. Disse ordrer er små ord, ofte forkortelser, der næsten alle kræver et eller flere parametre.
Mnemonics skrives som i et hvert andet programmeringssprog, blot må der kun være een mnemonic på hver linie.
Et eksempel på en meget almindelig mnemonic er 'ADD'. Denne mnemonic bruges selvfølgelig til at lægge 2 tal sammen, nærmere betegnet de 2 tal programmøren skriver bag ved 'ADD'. Eksempel: 'ADD #2,10'.

Før vi går videre med vore mnemonics, er det vigtigt at vi får defineret et par ting, omkring processoren og hukommelsen i computeren. Det er nemlig disse 2 elementer der er grundlaget for al programmering.

AMIGA'ENS PROCESSOR

Processoren, hjernen, i Amiga computeren hedder MC68000, og er fremstillet af det amerikanske selskab; Motorola Inc. Denne MC68000 er hvad de fleste kender som en 'chip' eller en 'lakrids'. Altså en sort aflang dims mèd en masse små ben på hver side, som er en fast ingrediens i alle computere og andre avancerede elektroniske apparater.
Det skal med det samme bekræftes at der i Amiga 3000 computeren sidder en MC68030 processor. Denne chip er blot en kraftigere og hurtigere version af MC68000, og kan programmeres med de samme mnemonics.

Uden at vi skal kaste os ud i store tekniske forklaringer om MC68000, er det dog på sin plads at vi her fortæller et par grundlæggende facts.
Der er i MC68000 en samling registre, hvor de fleste af dem kan påvirkes af programmøren. De 14 vigtigste kaldes;

D0, D1, D2, D3, D4, D5, D6, D7
og A0, A1, A2, A3, A4, A5, A6 og A7 / SP.

De 7 'D' registre er såkaldte Data-registre og de 7 'A' registre er Adresse-registre.
Hvert register repræsenterer en hukommelsesplads i processoren (MC68000) med 32 bits. Det vil sige at der i hvert register kan være et hvilket som helst tal, mellem 0 og 4.294.967.295.
Disse registre er tilgængelige for programmøren, og bruges sammen med næsten alle mnemonics. Når programmøren lægger en værdi ind i et register, bliver den der indtil den slettes igen. Derfor kan et register betegnes som en 'hukommelse'.

HUKOMMELSEN

Nu er processorens hukommelse sjældent stor nok til afvikling af selv små programmer. Derfor har computeren noget der hedder RAM (Random Access Memory), som er eet stort hukommelsesområde.
En Amiga 500 har 512 kilobytes hukommelse, hvilket kan opmåles således:

Et register i processoren består af 32 bits. Det samme princip bruges i RAM, blot er hukommelsen ikke delt op i 32-bits registre.
RAM er delt op i 'adresser', der starter med adresse nr. 0 og stiger indtil slutningen på den sidste RAM-chip i computeren. Hver adresse indeholder 8 bits, og kan indeholde et tal mellem 0 og 255. En 8-bits hukommelse kaldes i fagsproget for en 'byte'.
Inden for andre måleenheder bruges ordet 'kilo' i sammensætning med den aktuelle måleenhed, som f.eks. Kilometer og Kilojoule. Denne 'kilo' betegnelse findes også inden for måleenheden 'bytes', men hvor 'Kilo'

normalt betyder '*1000', betyder det i computer sammenhæng '*1024'. Det vil sige at 1 'kilobyte' er det samme som 1024 'bytes'. Derfor kan vi regne ud at en Amiga 500 med 512 kilobyte RAM, har en hukommelse på 512 * 1024 bytes = 524.288 bytes. Da hver byte indeholder 8 bits, vil det så sige at der ialt er 4.194.304 bits i en Amiga 500.

Resume:
En Amiga 500's hukommelse kan måles som : 512 kilobytes * 1024 bytes = 524.288 adresser, der hver kan indeholde et tal mellem 0 og 255.

Denne RAM hukommelse er altså stedet hvor man lagrer data, som processoren skal bruge i løbet af en program-afvikling.

For at vende tilbage til vores mnemonics, vil der her komme det første assembler program-eksempel:

```
MOVE    #3,D4       ;Flyt tallet 3 ind i register D4
ADD     #6,D4       ;Læg tallet 6 til indholdet i register D4
MOVE    D4,200000   ;Kopier indholdet af register D4 til adresse 200000
RTS                 ;Stop programmet
```

Slutresultatet af dette program er, at BÅDE register D4 og adresse 200000 indeholder tallet 9. Bemærk at vi kun kopierede register D4 til adresse 200000, og D4's indhold altså ikke blev slettet, men blot kopieret.

Som det kan ses, startes en assembler linie altid med en mnemonic. Derefter kommer så parametrene og hvis man ønsker det, kan man bagved sætte en kommentar til hvad der sker ved denne ordre. Læg mærke til den sidste mnemonic 'RTS', som er et eksempel på en mnemonic der ikke behøver noget parameter, da den blot stopper programmet.

START AF PROGRAMMER

Når der står et assembler program på skærmen, skal der tages meget få skridt før programmet kan startes.
Først skal program-teksten assembleres, hvilket gøres ved at vælge menupunktet 'Assemble' under menuen

'Assemble'. Med mindre der er fejl i programteksten, vil assembleren nu oversætte programmet til maskinkode. Når det er gjort, vælges 'Run' under 'Assemble' menuen, og maskinkodeprogrammet vil blive startet.

BEVÆGELSE I TEKSTEN

Bevægelse i programteksten foregår med piletasterne, samt Control og Shift tasterne.

Venstre	*Markør til venstre*
Højre	*Markør til højre*
Op	*Markør op*
Ned	*Markør ned*
SHIFT + venstre	*Markør til start af linie*
SHIFT + højre	*Markør til slutning af linie*
SHIFT + op	*Markør 1 side op*
SHIFT + ned	*Markør 1 side ned*
CONTROL + op	*Markør til start af tekst*
CONTROL + ned	*Markør til slutning af tekst*

FILES MENUEN

LOAD

(A + L)
Pack on/off
Tænder og slukker den automatisk udpakning af filer der hentes fra disken.
Load text
Henter en programtekst fra disken.
Load data
Henter en datafil fra disken ind på en absolut adresse. Der skal angives en start adresse og en længde.

SAVE

(A + S)
Pack on/off
Tænder og slukker den automatiske pakning af filer, der gemmes på disketten.
Save text
Gemmer den nuværende programtekst på disketten.
Save Paste buffer
Gemmer den sidst kopierede tekst på disketten.
Save data
Gemmer et hukommelsesområde som datafil på disketten. Der skal angives en start adresse og en længde.
Save object file
Gemmer et assembleret program på disken. Programmet kan eksekveres fra Workbench.

CLEAR

Clear text buffer
Sletter den nuværende programtekst.
Clear Paste buffer
Sletter den sidst kopierede tekst.
Clear Identifiers
Sletter identifier buffer.
Clear code buffer
Sletter assembleret kode.

DELETE

Delete file
Sletter en specifik fil på disketten.

PRINT

Print text buffer
Udskriver den nuværende programtekst til printeren.
Print Paste buffer
Udskriver den sidst kopierede tekst til printeren.

ABOUT

Viser hvem der har lavet InterAsm, samt serienummeret der skal bruges ved registrering.

QUIT

(A + Q)
Forlader InterAsm.

EDITOR MENUEN

MARK BLOCK

(Ctrl + B)
Starter markeringen af en tekstblok.

COPY BLOCK

(Ctrl + C)
Kopierer en markeret blok til Paste bufferen.

CUT BLOCK

(Ctrl + X)
Klipper en markeret blok til Paste bufferen.

PASTE BLOCK

(Ctrl + P)
Indsætter Paste bufferen's tekst i teksten, hvor markøren befinder sig.

SEARCH

(Ctrl + S)
Søger efter en given streng i programteksten.

REPLACE

(Ctrl + R)
Finder og erstatter strenge i programteksten.

ASSEMBLE MENUEN

ASSEMBLE

(A + A)
Assemblerer programteksten til maskinkode.

RUN

(A + R)
Starter det assemblerede program. En start adresse skal gives.

DISASSEMBLE

(A + D)
Disassembler et stykke hukommelse. En start adresse skal gives.

SINGLESTEP

On/off tænder og slukker for singlestep funktionen. Desuden kan det indstilles hvor store 'skridt' singlestepperen skal tage.

WATCH

On/off tænder og slukker for Watch funktionen. En adresse kan angives, samt om det er en byte, et word eller et longword der skal overvåges. Så snart InterAsm opdager en ændring i denne adresse, kommer der et vindue hvor det nye indhold af adressen vises.

SPECIAL MENUEN

CALCULATOR

(A + C)
Angiv et regneudtryk og få resultatet i decimal, hexadecimal og binær.

SHOW MEMORY

(A + M)
Angiv start adresse og se hukommelsen udskrevet i hexadecimal og ASCII.

FILL MEMORY

En start adresse, en slutadresse og et indhold skal angives. Derefter fyldes det angivne hukommelsesområde med det valgte indhold.

COPY MEMORY

Angiv en start adresse, en destinations adresse og en længde, hvorefter hukommelsen vil blive kopieret.

SEARCH MEMORY

Angiv en start adresse, en slut adresse og et datasæt, hvorefter InterAsm søger hukommelsen igennem efter de ønskede data.

MEMORY INFO

Her oplyses der om hvor InterAsm har sin maskinkode buffer, sin programtekst buffer, sin paste buffer og sin identifier buffer.

OPTIONS MENUEN

EDITOR

Her kan der bestemmes hvilket udseende cursoren skal have, samt hvorvidt den skal blinke eller ej.

ASSEMBLER

Short Branches optimerer automatisk assembleret kode, m.h.t. hop instruktioner.
Quick Instructions sørger for at alle muligheder for Quick instruktioner automatisk udnyttes.
JMP, JSR to BRA, BSR
Omformer automatisk JMP og JSR instruktioner til BRA og BSR instruktioner hvis muligt.

Såfremt De er fortrolig med maskinkodeprogrammering fra andre assemblere, vil De hurtigt erfare, hvor fleksibel InterAsm er med hensyn til syntaxen af kildeteksten. Dels vil De kunne bruge programmer skrevet på de fleste andre assemblere på markedet, og dels vil De uden større problemer kunne benytte den syntax og de assembler direktiver, De er vant til. Følgende oversigt viser, hvilke assembler direktiver der er tilgængelige i InterAsm og syntaxen for disse.

KOMMENTARER

Hvis assembleren støder på et semikolon eller et backslash (\), betragtes al efterfølgende tekst på linien som kommentarer og vil følgelig ikke blive assembleret.

TEKSTER

Tekster angives ved hjælp af apostrof eller gåseøjne og vil blive gemt som sekvenser af bytes. Dog kan de også benyttes som konstanter, hvor de vil antage ASCII-værdien af det første tegn i teksten. Eksempler:
DC.B “Dette er en test”,0
CMP.B #´a´,D4 ; Er D4=”a” (ASCII-kode 97)

TAL

InterAsm benytter 32-bits heltal, der kan angives som decimaltal, hexadecimale tal (prefix $) og binære tal (prefix %).
Eksempler: 1024 = $400 = %10000000000

REGNEUDTRYK

Regnudtryk kan bestå af tal, konstanter og labels samt de sædvanlige regneoperationer (,), +, -, *, og / (heltalsdivision).

LABELS

InterAsm godtager labels både med og uden efterfølgende kolon.

KONSTANT ERKLÆRINGER

InterAsm følger de to udbredte standarder

konstantnavn EQU regneudtryk og
konstantnavn = regneudtryk,

hvor labels og konstanter optrædende i regneudtrykket skal være defineret tidligere i kildeteksten.

ORG- OG LOAD-DIREKTIVER

Normalt anbringes den assemblerede kode i code-bufferen, men med LOAD-direktivet kan De specificere en anden startadresse, hvorfra koden rent fysisk vil blive lagt (Dette er dog kun tilrådeligt, såfremt de har allokeret plads til koden andetsteds i hukommelsen). Med ORG-direktivet kan De angive en fiktiv startadresse for koden, så programmet vil blive assembleret SOM OM det lå på adressen angivet af ORG, skønt det FAKTISK bliver anbragt på adressen angivet af LOAD. Hvis programmet skal udføres med det samme ved hjælp af RUN-kommandoen i assemble-menuen, skal LOAD og ORG (såfremt disse benyttes) angive samme adresse.

INCLUDE FILER

Med InterAsm er det muligt at inkludere kode fra andre tekstfiler ved hjælp af INCLUDE-direktivet. Teksten vil blive hentet ind løbende under assembleringen og kompileret på det sted, hvor INCLUDE-kommandoen står.
Syntax er:
INCLUDE “filnavn”

BYTES, WORDS OG LONGWORDS

Definering af enkelte bytes, words og longwords kan enten ske ved hjælp af direktiverne DB (define byte), DW (define word) og DL (define longword) eller DC (define constant) efterfulgt af .B, .W eller .L. Ved definering af sammenhængende blokke af data, kan De benytte enten BLK (blank) eller DS (define storage) efterfulgt af .B, .W eller .L samt antal reservationer af den pågældende længde, De ønsker, eventuelt efterfulgt af den værdi de gågældende bytes, words eller longwords skal antage.
Eksempel:
BLK.W 250,17 ; Reserver plads til 250 words med værdien 17

EVEN OG ODD

Da 68000-processoren kun kan udføre ordrer på lige adresser, er det muligt at tvinge InterAsm til at assemblere en kommando på en lige adresse ved hjælp af EVEN-direktivet. Ligeledes er det muligt at lægge data (af byte længde) på en ulige adresse ved hjælp af ODD-direktivet.

APPENDIX A : ASCII KODER

Dec.	Hex.	ASCII
000	00	
001	01	
002	02	
003	03	
004	04	
005	05	
006	06	
007	07	
008	08	
009	09	
010	0a	
011	0b	
012	0c	
013	0d	
014	0e	
015	0f	
016	10	
017	11	
018	12	
019	13	
020	14	
021	15	
022	16	
023	17	
024	18	
025	19	
026	1a	
027	1b	
028	1c	
029	1d	
030	1e	
031	1f	
032	20	
033	21	!
034	22	"
035	23	#
036	24	$
037	25	%
038	26	&

039	27	'
040	28	(
041	29	)
042	2a	*
043	2b	+
044	2c	,
045	2d	-
046	2e	.
047	2f	/
048	30	0
049	31	1
050	32	2
051	33	3
052	34	4
053	35	5
054	36	6
055	37	7
056	38	8
057	39	9
058	3a	:
059	3b	;
060	3c	<
061	3d	=
062	3e	>
063	3f	?
064	40	@
065	41	A
066	42	B
067	43	C
068	44	D
069	45	E
070	46	F
071	47	G
072	48	H
073	49	I
074	4a	J
075	4b	K
076	4c	L
077	4d	M
078	4e	N
079	4f	O

APPENDIX A : ASCII KODER

080	50	P
081	51	Q
082	52	R
083	53	S
084	54	T
085	55	U
086	56	V
087	57	W
088	58	X
089	59	Y
090	5a	Z
091	5b	[
092	5c	\
093	5d	]
094	5e	^
095	5f	_
096	60	`
097	61	a
098	62	b
099	63	c
100	64	d
101	65	e
102	66	f
103	67	g
104	68	h
105	69	i
106	6a	j
107	6b	k
108	6c	l
109	6d	m
110	6e	n
111	6f	o
112	70	p
113	71	q
114	72	r
115	73	s
116	74	t
117	75	u
118	76	v
119	77	w
120	78	x

121	79	y
122	7a	z
123	7b	{
124	7c	|
125	7d	}
126	7e	~

APPENDIX B : MNEMONICS

Mnemonic	Operation
ABCD	Add Decimal with Extend
ADD	Add Binary
ADDA	Add Address
ADDI	Add Immediate
ADDQ	Add Quick
ADDX	Add Extended
AND	AND Logical
ANDI	AND Immediate
ASL, ASR	Arithmetic Shift
Bcc	Branch Conditionally
BCHG	Test a Bit and Change
BCLR	Test a Bit and Clear
BRA	Branch Always
BSET	Test a Bit and Set
BSR	Branch to Subroutine
BTST	Test a Bit
CHK	Check Register Against Bounds
CLR	Clear an Operand
CMP	Arithmetic Compare
CMPA	Arithmetic Compare Address
CMPI	Compare Immediate
CMPM	Compare Memory
DBcc	Test Condition
DIVS	Signed Divide
DIVU	Unsigned Divide
EOR	Exclusive OR Logical
EORI	Exclusive OR Immediate
EXG	Exchange Registers
EXT	Sign Extend
JMP	Jump
JSR	Jump to Subroutine
LEA	Load Effective Address
LINK	Link and Allocate
LSL, LSR	Logical Shift
MOVE	Move Data from Source to Destination
MOVE from SR	Move from the Status Register

MOVE to CCR	Move to Condition Codes Register
MOVE to SR	Move to the Status Register
MOVE USP	Move User Stack Pointer
MOVEA	Move Address
MOVEM	Move Multiple Registers
MOVEP	Move Peripheral Data
MOVEQ	Move Quick
MULS	Signed Multiply
MULU	Unsigned Multiply
NBCD	Negate Decimal with Extend
NEG	Two's Complement Negation
NEGX	Negate with Extend
NOP	No Operation
NOT	Logical Complement
OR	Inclusive OR Logical
ORI	Inclusive OR Immediate
PEA	Push Effective Address
RESET	Reset External Devices
ROL, ROR	Rotate without Extend
ROXL, ROXR	Rotate with Extend
RTE	Return from Exception
RTR	Return and Restore Condition Codes
RTS	Return from Subroutine
SBCD	Subtract Decimal with Extend
Scc	Set according to Condition
STOP	Stop Program Execution
SUB	Subtract Binary
SUBA	Subtract Address
SUBI	Subtract Immediate
SUBQ	Subtract Quick
SUBX	Subtract with Extend
SWAP	Swap Register Halves
TAS	Test and Set an Operand
TRAP	Trap
TRAPV	Trap on Overflow
TST	Test an Operand
UNLK	Unlink

Constant expected
Et absolut tal var forventet blandt parametrene.

Overflow
Det absolutte tal i denne kommando's parametre er for stort (større end 32 bit).

Nonsense statement
En semantisk fejl er opstået. Der er rod i denne linie.

"," expected
Der mangler et komma.

Syntax error
Der er en stavefejl i denne linie.

" expected
Der mangler en "

Identifier too long
Labels må max. være 30 karakterer lange.

Text too long
Denne tekstlinie er for lang. Der må max. være 256 karakterer på en linie.

Identifier allready declared
Denne label er allerede brugt tidligere i programmet.

End Of Line expected
Der er overskydende tekst efter kommandoen og parametrene. Linien skal deles, eller den overskydende tekst skal defineres som kommentar.

Command expected
Det første tegn/ord efter en evt. label er ikk e en kommando.

Bad size qualifier
Fejl i angivelse af .B, .W eller .L

A-register expected
Et adresse-register (A0-A7) er forventet blandt parametrene.

")" expected
Der mangler en højre parentes.

PC or A-register expected
Et adresse-register (A0-A7) eller PC registeret er forventet blandt parametrene.

D- or A-register expected
Et D-register (D0-D7) eller et A-register (A0-A7) er forventet blandt parametrene.

Wrong adressing mode
Der er benyttet en ugyldig adresserings måde for denne ordre.

Out of range
Tal er for stort eller lille til ordren.

"#" expected
Der mangler en "#" blandt parametrene.

Odd adress
Ordren ligger på en ulige adresse. Udlign med en EVEN.

Identifier not declared
En label blandt parametrene er ikke defineret.

BRUGEREN HAR INDFLYDELSE PÅ SENERE VERSIONER

Vi håber at de vil få glæde af Deres produkt. Skulle programmet efter Deres vurdering lide under mangler, hører vi meget gerne derom.

- *Er programmet for langsomt ?*
- *Er manualen ubrugelig ?*
- *Mangler der en vigtig funktion ?*
- *Er kassen uhåndterlig ?*

Har De et eller flere ankepunkter, er De velkommen til at kontakte Interactivision, **skriftligt.**
Vi gennemlæser alle henvendelser og bestræber os på at medtage så mange som muligt, relevante forbedringer i næste opgradering.

NYE PROJEKTER OG IDEER ER ALTID VELKOMNE HOS OS

Kan De selv programmere? Mange hjemmeprogrammører har tit rigtig gode programmer liggende i skuffen, eller også har de evnerne og ideerne til at udvikle et succesrigt stykke software.
Interactivision er altid åbne for spændende mennesker med spændende ideer.
De er med dette produkt blevet bekendt med Interactivisions standard, og har måske allerede vurderet om det er et niveau, hvor De også kan være med.
Skriv til os, og fortæl om Dem selv og Deres ideer. Vi garanterer naturligvis den største diskretion, specielt hvad angår medsendte program eksempler.

Med venlig hilsen

Interactivision ApS.
Hovedgårdsvej 4
DK-8600 SIlkeborg

This was brought to you

from the archives of

http://retro-commodore.eu